CURS 3

CSS. Stilizare. Layout. Design Responsiv

3.1 Cum Funcționează CSS

CSS (Cascading Style Sheets) descrie prezentarea vizuală a HTML. Cuvântul 'cascading' se referă la algoritmul care determină ce regulă câștigă când mai multe reguli se aplică aceluiași element.

Cele 3 moduri de aplicare CSS - ordinea specificității

Tip	Sintaxă	Recomandare
Extern (recomandat)	<link rel="stylesheet" href="style.css">	Separare completă HTML/CSS. Reutilizabil pe mai multe pagini. ÎNTOTDEAUNA preferat.
Intern (evitat)	<style> p { color: red; } </style> în <head>	Util pentru stiluri specifice unei pagini sau pentru prototipare rapidă. Nu reutilizabil.
Inline (interzis practic)	<p style="color: red;">	Cel mai înalt specificity, greu de menținut, nu se poate suprascrie ușor. Evitat în producție.

Selectori CSS - de la simplu la complex

```
/* Element - 1 punct specificitate */
p { color: #333; }

/* Clasă - 10 puncte */
.card { border-radius: 8px; }

/* ID - 100 puncte (evitat pentru stilizare) */
#header { background: navy; }

/* Atribut - 10 puncte */
input[type="email"] { border-color: blue; }

/* Pseudo-clasă - 10 puncte */
a:hover { color: orange; }
input:focus { outline: 2px solid blue; }
li:nth-child(odd) { background: #f5f5f5; }

/* Pseudo-element - 1 punct */
p::first-line { font-weight: bold; }
h2::before { content: '→ '; }

/* Combinatori */
.card .titlu { font-size: 1.25rem; }
.nav > li { display: inline; }
h2 + p { margin-top: 0; }
h2 ~ p { color: gray; }
```

> Specificitate: o regulă cu ID (#) va câștiga întotdeauna contra unei reguli cu clasă (.card), chiar dacă regula cu clasă vine mai târziu în fișier. Dacă ai nevoie să suprascrii stiluri și nu funcționează, verifică specificitatea, nu adăuga !important.

3.2 Box Model - Fundamentul Layout-ului CSS

Fiecare element HTML este redat ca un dreptunghi. Înțelegerea straturilor acestui dreptunghi este esențială pentru orice layout.

Straturile box model-ului (din interior spre exterior)

Strat	Proprietăți CSS	Comportament
Content	width, height	Zona cu conținut efectiv (text, imagine). Dimensiunile implicite se calculează din conținut.
Padding	padding, padding-top/right/bottom/left	Spațiu între conținut și border. Are culoarea de fundal a elementului. Adaugă la dimensiune (implicit).
Border	border, border-width/style/color	Linie în jurul padding-ului. Vizibilă sau invizibilă. Adaugă la dimensiune (implicit).
Margin	margin, margin-top/right/bottom/left	Spațiu transparent în exteriorul border-ului. Nu are culoare. Separă elementele între ele.

box-sizing: border-box - regula de aur

Implicit, width și height se referă DOAR la zona de content. Padding și border se ADAUGĂ pe deasupra. Aceasta provoacă layout-uri neașteptate.

```
/* ✗ Comportament implicit (content-box): */
/* width: 200px + padding: 20px + border: 2px = 244px TOTAL */

/* ☑ ÎNTOTDEAUNA adaugă aceasta la începutul oricărui fișier CSS: */
*, *::before, *::after {
  box-sizing: border-box;
}
/* Acum: width: 200px include padding și border. */
/* width: 200px + padding: 20px + border: 2px = 200px TOTAL */
```

Margin collapse - comportament unic al margin-urilor

Margin-urile verticale (top/bottom) între elemente BLOCK adiacente se contopesc - valoarea mai mare câștigă:

```
h2 { margin-bottom: 24px; }
p  { margin-top: 16px; }
/* Spațiul dintre h2 și p = 24px (nu 40px!) */

/* Soluție: folosește margin doar pe o direcție (de preferință jos) */
/* sau folosește gap în Flexbox/Grid (nu are collapse) */
```

3.3 Flexbox - Layout Unidimensional

Flexbox organizează elementele pe o singură axă (orizontală SAU verticală). Este ideal pentru: bare de navigație, carduri într-un rând, centrare verticală/orizontală, distributie uniformă a spațiului.

```
.container-flex {
  display: flex;              /* activează flexbox pe container */
  flex-direction: row;        /* row | column | row-reverse | column-reverse */
  justify-content: space-between; /* distribuie pe axa principală */
  align-items: center;        /* aliniază pe axa secundară */
  flex-wrap: wrap;            /* permite trecerea pe linie nouă */
  gap: 16px;                  /* spațiu între itemi (nu margin-uri!) */
}

/* Pe fiecare item flex */
.item-flex {
  flex: 1;                    /* prescurtare: flex-grow flex-shrink flex-basis */
  /* flex: 1 = 'ocupă spațiu disponibil proporțional' */
  /* flex: 0 0 200px = 'exact 200px, nu crește, nu se micșorează' */
  align-self: flex-start;     /* suprascrie align-items pentru acest item */
  order: -1;                  /* muta item-ul fără a modifica HTML-ul */
}
```

Proprietăți Flexbox - tabel de referință

Proprietate (pe container)	Valori și efect
justify-content	flex-start | flex-end | center | space-between (spațiu între) | space-around (spațiu în jur) | space-evenly (spațiu egal) - pe axa principală
align-items	stretch (implicit - înalțimea egalată) | flex-start | flex-end | center | baseline - pe axa secundară
align-content	Ca justify-content dar pentru linii multiple (când flex-wrap: wrap). Funcționează doar cu mai multe linii.
flex-direction	row (stânga→dreapta) | row-reverse | column (sus→jos) | column-reverse
flex-wrap	nowrap (implicit - toți pe o linie) | wrap (trece pe linie nouă) | wrap-reverse
gap / row-gap / column-gap	Spațiu între itemi. Înlocuiește tehnicile vechi cu margin. Nu are collapse.

> Exercițiu recomandat: Flexbox Froggy (https://flexboxfroggy.com/) - 24 de nivele interactive pentru stăpânirea Flexbox în 30 de minute. Completează-l înainte de Laboratorul 3.

3.4 CSS Grid - Layout Bidimensional

CSS Grid organizează elementele pe două axe simultan (rânduri ȘI coloane). Este ideal pentru: layout-uri de pagini complete, galerii de imagini, dashboard-uri, orice design cu rânduri și coloane.

```
.grid-container {
  display: grid;

  /* Definim coloanele: 3 coloane, prima fixă, ultimele două flexibile */
  grid-template-columns: 280px 1fr 2fr;
```

```
    /* fr = fracțiune din spațiul disponibil */

    /* Rânduri implicite de 200px minim, 1fr maxim */
    grid-auto-rows: minmax(200px, 1fr);

    /* Repetare: 4 coloane egale */
    /* grid-template-columns: repeat(4, 1fr); */

    /* Auto-fill: atâtea coloane cât încap de min 250px */
    /* grid-template-columns: repeat(auto-fill, minmax(250px, 1fr)); */

    gap: 24px 16px;   /* row-gap column-gap */
  }

  /* Plasarea explicită a itemilor */
  .item-mare {
    grid-column: 1 / 3;   /* de la linia 1 la linia 3 (span 2 coloane) */
    grid-row: 1 / 3;      /* de la linia 1 la linia 3 (span 2 rânduri) */
  }

  /* Scurtătură: grid-area */
  .item-mare { grid-area: 1 / 1 / 3 / 3; }
  /* Format: row-start / col-start / row-end / col-end */
```

grid-template-areas - Layout cu nume

Această proprietate permite descrierea layout-ului vizual direct în CSS:

```
  .pagina {
    display: grid;
    grid-template-columns: 220px 1fr;
    grid-template-rows: 64px 1fr 48px;
    grid-template-areas:
      "header  header"
      "sidebar main  "
      "footer  footer";
    min-height: 100vh;
  }

  header  { grid-area: header; }
  nav     { grid-area: sidebar; }
  main    { grid-area: main; }
  footer  { grid-area: footer; }
```

> CSS Grid Garden (https://cssgridgarden.com/) - 28 de nivele interactive pentru stăpânirea Grid. Completează-l după Flexbox Froggy pentru o bază solidă de layout.

3.5 Design Responsiv

Design-ul responsiv înseamnă că interfața se adaptează automat la dimensiunea ecranului dispozitivului - de la telefon (320px) la monitor 4K (3840px).

Unități relative - baza design-ului fluid

Unitate	Se calculează față de	Utilizare tipică
%	Elementul părinte	Lățimi fluide: width: 90%

em	Font-size al elementului curent	Padding, margin relativ la text: padding: 1.5em
rem	Font-size al elementului rădăcină (html)	Tipografie, spații consistente la nivel de pagină: font-size: 1.125rem
vw	1% din lățimea viewport-ului	Secțiuni full-width: width: 100vw
vh	1% din înălțimea viewport-ului	Secțiuni full-height: height: 100vh
min()	Cel mai mic din valorile date	Tipografie responsivă: font-size: min(4vw, 2rem)
max()	Cel mai mare din valorile date	Lățime minimă: width: max(300px, 50%)
clamp()	Valoare între un minim și maxim	font-size: clamp(1rem, 2.5vw, 1.5rem)

Media Queries

Media queries aplică stiluri CSS condiționat, în funcție de caracteristicile dispozitivului:

```
/* Abordare Mobile-First (RECOMANDATĂ) */
/* Stilurile de bază se scriu pentru mobile, fără media query */

.nav-lista {
  display: flex;
  flex-direction: column;  /* vertical pe mobile */
  gap: 8px;
}

/* Tabletă și mai mare */
@media (min-width: 768px) {
  .nav-lista {
    flex-direction: row;   /* orizontal pe tabletă+ */
    gap: 24px;
  }
}

/* Desktop */
@media (min-width: 1024px) {
  .container {
    max-width: 1200px;
    margin-inline: auto;
  }
}

/* Preferință sistem dark mode */
@media (prefers-color-scheme: dark) {
  :root { --fundal: #1a1a1a; --text: #e0e0e0; }
}

/* Preferință animații reduse (accesibilitate) */
@media (prefers-reduced-motion: reduce) {
  * { animation-duration: 0.01ms !important; }
}
```

> Mobile-First vs Desktop-First: Mobile-First (min-width) este abordarea recomandată modern. Înseamnă că stilurile de bază sunt pentru ecrane mici, și adăugăm complexitate pe ecrane mai mari. Este mai ușor de scalat și performant.

3.6 CSS Custom Properties (Variabile)

Variabilele CSS permit definirea valorilor o singură dată și reutilizarea lor în tot fișierul. Modificând o variabilă, schimbi automat toate aparițiile ei.

```
/* Definire în :root - disponibile global */
:root {
  --culoare-primara: #1B3A6B;
  --culoare-accent: #E87722;
  --font-titlu: 'Space Grotesk', sans-serif;
  --sp-md: 16px;
  --raza: 8px;
  --umbra: 0 4px 12px rgba(0,0,0,0.1);
}

/* Utilizare */
.buton-primar {
  background: var(--culoare-accent);
  padding: var(--sp-md);
  border-radius: var(--raza);
  box-shadow: var(--umbra);
}

/* Suprascrierea locală (cascadă) */
.dark-theme {
  --culoare-primara: #90CAF9;   /* albastru deschis pentru dark mode */
  --fundal: #1a1a2e;
}

/* Manipulare prin JavaScript */
// document.documentElement.style
//   .setProperty('--culoare-accent', '#FF6B6B');
```

> Sfat profesionist: Folosește variabile CSS pentru tot ce este reutilizabil: culori, fonturi, dimensiuni, spații, raze, umbre, tranziții. Un sistem de design bun se definește în :root și se folosește exclusiv prin variabile. Niciodată nu hard-codezi o culoare de două ori.

Exemplu Practic în VS Code - DevConnect: CSS Complet

Continuăm proiectul DevConnect adăugând stilizarea CSS. Creează folderul css/ în proiect și fișierele descrise în Proiect-Model Lab 3 din Volumul I al acestor note.

Testare responsive în VS Code / Browser

Live Server deschide pagina în browser. Pentru a testa responsivitatea:

1. Deschide DevTools (F12)
2. Apasă iconița 'Toggle device toolbar' (sau Ctrl+Shift+M)
3. Selectează un preset (iPhone SE = 375px, iPad = 768px, etc.) sau introdu manual lățimea
4. Verifică că layout-ul se comportă corect la fiecare breakpoint
5. Testează și cu orientare landscape (butonul de rotire)

Referințe pentru Curs 3

★ CSS-Tricks - Ghid complet Flexbox
★ CSS-Tricks - Ghid complet Grid
★ web.dev - Curs CSS Google (gratuit)
Flexbox Froggy - joc interactiv Flexbox
Grid Garden - joc interactiv CSS Grid
MDN CSS - Referință completă
Can I Use - compatibilitate browsere pentru proprietăți CSS
CSS Specificity Calculator

Notă privind elaborarea materialelor de curs

Vreau să fiu transparent cu voi: structura și conținutul acestor note de curs au fost generate cu ajutorul unui instrument de inteligență artificială (Claude, de la Anthropic), pe baza cerințelor și direcțiilor pe care le-am formulat eu ca titular de curs.

De ce vă spun asta? Pentru că:
• Nu pot garanta că fiecare noțiune tehnică are 100% acuratete sau este actualizată
• Vă încurajez să verificați activ sursele bibliografice indicate
• Utilizarea responsabilă a AI în educație înseamnă transparență, nu ascundere

Considerați aceste materiale un ghid structurat de studiu, nu un manual definitiv. Dacă identificați o eroare sau o neclaritate, veniți cu ea la curs